# ­­­Login Screen

The user is required to log in with either Facebook or Google+.

## Provider login

At login the app will make a call to the server with the email of the user that has logged in. The return result will be a JSON with key value pairs described in the tables below. Included in this JSON is the field “isProvider”. If this is set to true that means the user is a provider. If it is false and this is a new account, prompt the user with the following choice.

Are you a service provider?

If yes, continue. If no go to section 2.

When the user is complete a JSON shall be posted to the same URL with the updated information.

GET /user?email=<email-address> - Lookup user info.

PUT /user - Add a new user. Returns response {“userId”, <Long ID>} on success.

The field “complete” will be set to “true” if the account exists and is complete. If the account has been created and is incomplete, or the account has not been created this value will be set to “false”.

The provider user settings screens will be used in the following cases.

1. A new account needs to be created at login.
2. An account exists but is incomplete.
3. User edits account from the provider dashboard.

### Provider Account Does Not Exist

Prompt for the following information. The tables below shall be displayed in separate screens with navigation buttons to go to a previous screen.

#### User Information Input

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Name | Data type expected | Required | Input JSON Filed Name | Notes |
| Email |  |  | email | Use the value from login. This is not a user input but should be displayed. |
| Email Address | Str | Yes | email | Use the email from login. |
| First Name | Str | Yes | firstName |  |
| Last Name | Str | Yes | lastName |  |
| Business Name | Str | no | businessName |  |
| Business Address | String | Yes | address |  |
| City | str | Yes | city |  |
| State | str | Yes | state |  |
| Zip code |  |  | zip |  |
| Phone ok | Boolean | Defaults true | phoneOk | Checkbox |
| Email ok | Boolean | Default true | emailOk | Checkbox |
| Text ok | Boolean | Defaults true | textOk | Checkbox |
|  |  |  |  |  |

#### Provider License

A provider can have multiple licenses, but must have at least one. The “licenses” field in the input JSON will be a list of JSONS.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data type expected | | Required Input | Input JSON Field Name | Notes |
| License Number | Alpha numeric | yes | | licenseNumber |  |
| Service Type | String | yes | | serviceType | This list will be supplied by the server. For the initial release the options |
| Special Notes | String | no | | notes |  |

Once a license is added the user shall be prompted

“Would you like to add an additional license?”

If yes, repeat until finished.

#### Provider Availability

The following illustrates the input values received from the server, and the options to pass back to the server.

The UI shall follow these rules.

1. Days are clicked to enable / disable.
2. Include options to set the same time for weekdays, weekends, everyday.
3. Include check box for “Always Available.”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Name | Data type expected | Required | Input JSON Filed Name | Notes |
| Monday Start | HH:MM | no | mondayStart |  |
| Monday end | HH:MM | no | mondayEnd |  |
| Tuesday Start | HH:MM | no | tuesdayStart |  |
| Tuesday End | HH:MM | no | tuesdayEnd |  |
| Wednesday Start | HH:MM | no | wednesdayStart |  |
| Wednesday End | HH:MM | no | wednesdayEnd |  |
| Thursday Start | HH:MM | no | thursdayStart |  |
| Thursday End |  | no | thursdayEnd |  |
| Friday Start | HH:MM | no | fridayStart |  |
| Friday End | HH:MM | no | fridayEnd |  |
| Saturday Start | HH:MM | no | saturdayStart |  |
| Saturday End | HH:MM | no | saturdayEnd |  |
| Sunday Start | HH:MM | no | sundayStart |  |
| Sunday End | HH:MM | no | sundayEnd |  |

#### Additional JSON Field

The following fields will be included in the JSON. In the future this list may increase so no errors should be thrown when unrecognized fields are found.

The server will ignore these fields so the application is not required to set or include them.

|  |  |  |  |
| --- | --- | --- | --- |
| Field Name | Data type expected | Required Input | Notes |
| isProvider | Boolean |  |  |
| complete | true |  |  |
| userId | Long |  | Will be blank until the user is created. |

### Provider Account Exists and is Incomplete / Edit Account

Populate all the views described in the previous section with the information returned from the server. Once complete an UPDATE should be used instead of a POST.

POST <URI>/account - Update an existing user.

### Provider Dashboard

A placeholder dashboard should be created for a provider as their home page. The following buttons and actions should be available.

1. Edit Account
2. Search for provider.

## User Login

Go directly to the select service type screen.

# Provider locations

A background timer shall send the provider location every N minutes, where N is configurable for testing.

POST <URI>/location/<userId> {“longitude” : <float>, “latitude”, <float>}

# Service Query

The service query screen should be a scrollable listview of service types listed from the server. Each item will be an icon / image representative of the service type (Electrical would be a lighting bolt or a power plug, etc) and then the service type text.

The user selects one of the items to search for a service.

## Query Service Types

To get the available service types use the following query.

GET <URI>/services?includeIcon=[true, false]

The includeIcon will tell the server to include the icon in the result JSON as a base64 string.

In order to keep the bandwidth as low as possible, the icon query should be done a single time and cached while the application is running. In the event that the icon changed on the server and a user has a different version, it really doesn’t matter. The icon can be fetched when the application is restarted.

This will include a list of service type JSONs.

|  |  |  |
| --- | --- | --- |
| Field Name | Data Type Expected | Notes |
| serviceName | String | The service name. Display this string as is. |
| icon | String | Base64 string representing an icon. |

These icon values should be used for the service type when querying, when picking a service type for a license as well as the icons for ratings in service queries.

## Query Results

### REST Query

Once the service type is know make a REST call to the server with the users current location.

GET <URI>/provider/<type>?longitude=<float>&latitude=<float>&exclude=[providerIds]

The server will return a JSON with provider information.

***AN EXAMPLE JSON file will be provided.***

### Results Display

The results should be a scrollable listview with the results.

| Business Name| Distance | Rating |

When an item is selected it should expand to show all of the business information.

Required fields will be provide.

If the provider has the phoneOk flag set or the textOk set, the user can click on the number and a prompt to call or text.

The list of results should perform like an infinite scroll. When making the server call from 3.1.1 the exclude option should be the userIds of the providers already returned. An empty JSON result will be returned when there are no more providers to return.

# REST API

## Create new user

/